

request , response , session , application
() () ()

JSP (Java Sever Page)

- 기본 객체와 영역

- >

- >

/

01. JSP 내장객체 개요

■ JSP 내장객체란?

- JSP 내장객체란 'JSP 내에서 선언하지 않고 사용할 수 있는 객체'라는 의미에서 붙여진 이름.
- 구조적으로는 JSP가 서블릿 형태로 자동 변환된 코드 내에 포함되어 있는 멤버변수, 메서드 매개변수 등의 각종 참조 변수(객체)를 말함.

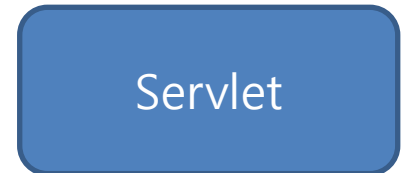
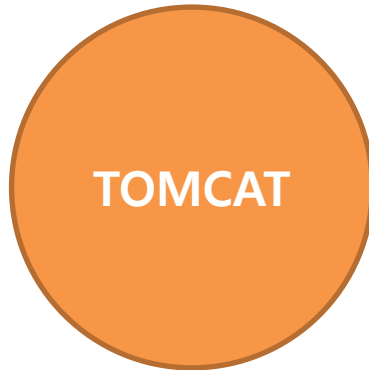
01. JSP 내장객체 개요

참조 변수 이름(내장객체)	자바 클래스	주요 역할
request	javax.servlet.http.HttpServletRequest	HTML 폼 요소의 선택 값 등 사용자 입력 정보를 읽으려고 사용한다.
response	javax.servlet.http.HttpServletResponse	사용자 요청에 대한 응답을 처리하려고 사용한다.
<u>pageContext</u>	javax.servlet.jsp.PageContext	현재 JSP 실행에 대한 context 정보를 참조하려고 사용한다.
session	javax.servlet.http.HttpSession	클라이언트의 세션 정보를 처리하려고 사용한다.
application	javax.servlet.ServletContext	웹 서버의 애플리케이션 처리와 관련된 정보를 참조하려고 사용한다.
out	javax.servlet.jsp.JspWriter	사용자에게 전달하기 위한 output 스트림을 처리하려고 사용한다.
config	javax.servlet.ServletConfig	현재 JSP의 초기화 환경을 처리하려고 사용한다.
page	java.lang.Object	현재 JSP의 클래스 정보를 보려고 사용한다.
exception	java.lang.Throwable	예외 처리를 하려고 사용한다.

01. JSP 내장객체 개요

1. JSP 내장객체의 구조적 특징

- 서블릿으로 변경된 JSP 코드는 모두 `_jspService()` 메서드에 위치함.
- 메서드 매개변수인 `request`, `response` 를 비롯한 `pageContext`, `session`, `application`, `page`, `config`, `out` 등 메서드 내에서 참조할 수 있는 참조변수들이 내장객체가 됨.



01. JSP 내장객체 개요

```
public void _jspService(HttpServletRequest request, HttpServletResponse response) throws java.io.IOException, ServletException {
```

```
    JspFactory _jspxFactory = null;
```

```
    javax.servlet.jsp.PageContext pageContext = null;
```

```
    HttpSession session = null;
```

```
    ServletConfig config = null;
```

```
    JspWriter out = null;
```

```
    Object page = this;
```

```
    JspWriter _jspx_out = null;
```

```
    try {
```

```
        pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
```

```
        application = pageContext.getServletContext();
```

```
        config = pageContext.getServletConfig();
```

```
        session = pageContext.getSession();
```

```
        out = pageContext.getOut();
```

```
        ...
```

```
    }
```

hello.jsp

```
<html>
```

```
<head> <title>Hello JSP</title> </head>
```

```
<body>
```

```
<h1> Hello JSP Test</h1>
```

```
<%out.println("<font color=blue>Hello World! JSP</font>");%>
```

```
</body> </html>
```

```
        out.println("<font color=blue>Hello World! JSP</font>");
```

```
        out.write("WrWn</body> </html> WrWnWrWnWrWn");
```

01. JSP 내장객체 개요

_jspService의 구성

매개변수2개

1. `HttpServletRequest request`
2. `HttpServletResponse response`

필요한 지역변수

1. `PageContext pageContext = null;`
2. `HttpSession session = null;`
3. `ServletContext application = null;`
4. `ServletConfig config = null;`
5. `JspWriter out = null;`
6. `Object page = this;`
7. `JspFactory _jspxFactory = null;`
8. `String _value = null;`

지역변수초기화

1. `_jspxFactory = JspFactory.getDefaultFactory();`
2. `pageContext = _jspxFactory.getPageContext(this, request, response, "", true, 8192, true);`
3. `application = pageContext.getServletContext();`
4. `config = pageContext.getServletConfig();`
5. `session = pageContext.getSession();`
6. `out = pageContext.getOut();`

01. JSP 내장객체 개요

2. 내장객체를 이용한 속성 관리 기법

- 내장객체가 단순히 특정한 기능을 제공하는 컨테이너 관리 객체라는 점 외에도 한 가지 특징이 있다. 바로 `page, request, session, application` 내장객체를 이용한 속성 관리 기법이다. 이들 내장객체는 각자 지정된 `생명주기`가 있으며 `setAttribute()`, `getAttribute()`라는 메서드를 통해 해당 생명주기 동안 자바 객체를 유지하는 기능을 제공한다.

(
String, data)
object

02. request

■ request 내장객체

- request는 사용자 요청과 관련된 기능을 제공하는 내장객체로 `javax.servlet.http.HttpServletRequest` 클래스에 대한 참조 변수임.
- 주로 클라이언트에서 서버로 전달되는 정보를 처리하기 위해 사용한다.
- 대표적으로 HTML 폼을 통해 입력된 값을 JSP에서 가져올 때 사용함.

02. request

메서드	String 가	설명	Iterator
getParameterNames()		현재 요청에 포함된 매개변수의 이름을 열거(Enumeration) 형태로 넘겨준다.	
* getParameter(name)		문자열 name과 이름이 같은 매개변수의 값을 가져온다.	
name userid 가			
getParameterValues(name)		문자열 name과 이름이 같은 매개변수의 값을 배열 형태로 가져온다. checkbox, multiple list 등에 주로 사용한다.	
* getCookies()		모든 쿠키 값을 javax.servlet.http.Cookie의 배열 형태로 가져온다.	
cookie[] data			
getMethod()		현재 요청이 GET이나 POST 형태로 가져온다.	
* getSession()		현재 세션 객체를 가져온다.	
getRemoteAddr()		클라이언트의 IP 주소를 알려준다.	
getProtocol()		현재 서버의 프로토콜을 문자열 형태로 알려준다.	
* setCharacterEncoding()		현재 JSP로 전달되는 내용을 지정한 캐릭터셋으로 변환해준다. HTML 폼에서 한글 입력을 정상적으로 처리해주려면 반드시 필요하다.	
("utf - 8")			

02. request

- [실습] request 테스트 폼(request_form.html)

가 - > 가
- >

request Form Test

form action=" " type

이름

직업 ▼

관심사항 java ☒ HTML5 ☒ css3 ☒ javascript ☒ JSP ☐

02. request

■ [실습] request 테스트 결과(request_result.jsp)

전송 결과

이름	test
직업	시스템엔지니어
관심사항	java HTML5 css3 javascript

03. response

■ response 내장객체

String 가

- response는 request와 반대되는 개념으로, 사용자 응답과 관련된 기능을 제공.

✖ 사용자 요청(request)을 처리하고 응답을 다른 페이지로 전달하는 등의 기능을 제공한다.

- javax.servlet.http.HttpServletResponse 객체에 대한 참조 변수로, request에 만큼 많이 사용되지는 않으나 `setContentType`, `sendRedirect`와 같은 메서드는 잘 알아두어야 한다.

메서드	설명
✖ <code>setContentType(type)</code>	문자열 형태의 type에 지정된 MIME Type으로 contentType을 설정한다.
<code>setHeader(name,value)</code>	문자열 name의 이름으로 문자열 value의 값을 헤더로 세팅한다.
<code>setDateHeader(name, date)</code>	문자열 name의 이름으로 date에 설정된 밀리세컨드 시간 값을 헤더에 설정한다.
<code>sendError(status,msg)</code>	오류 코드를 세팅하고 메시지를 보낸다.
✖ <u><code>sendRedirect(url)</code></u>	클라이언트 요청을 다른 페이지로 보낸다.

03. response

■ [실습] forward 액션과의 차이 알아보기(page_control.jsp)



response_sendRedirect.jsp page_control.jsp page_control.jsp

http://localhost:8080/jspbook/ch06/page_control.jsp

forward, sendRedirect 테스트

forward action :

response.sendRedirect :

03. response

- [실습] sendRedirect 메서드를 이용한 호출(response_sendRedirect.jsp), 호출된 화면(page_control_end.jsp)



04. out

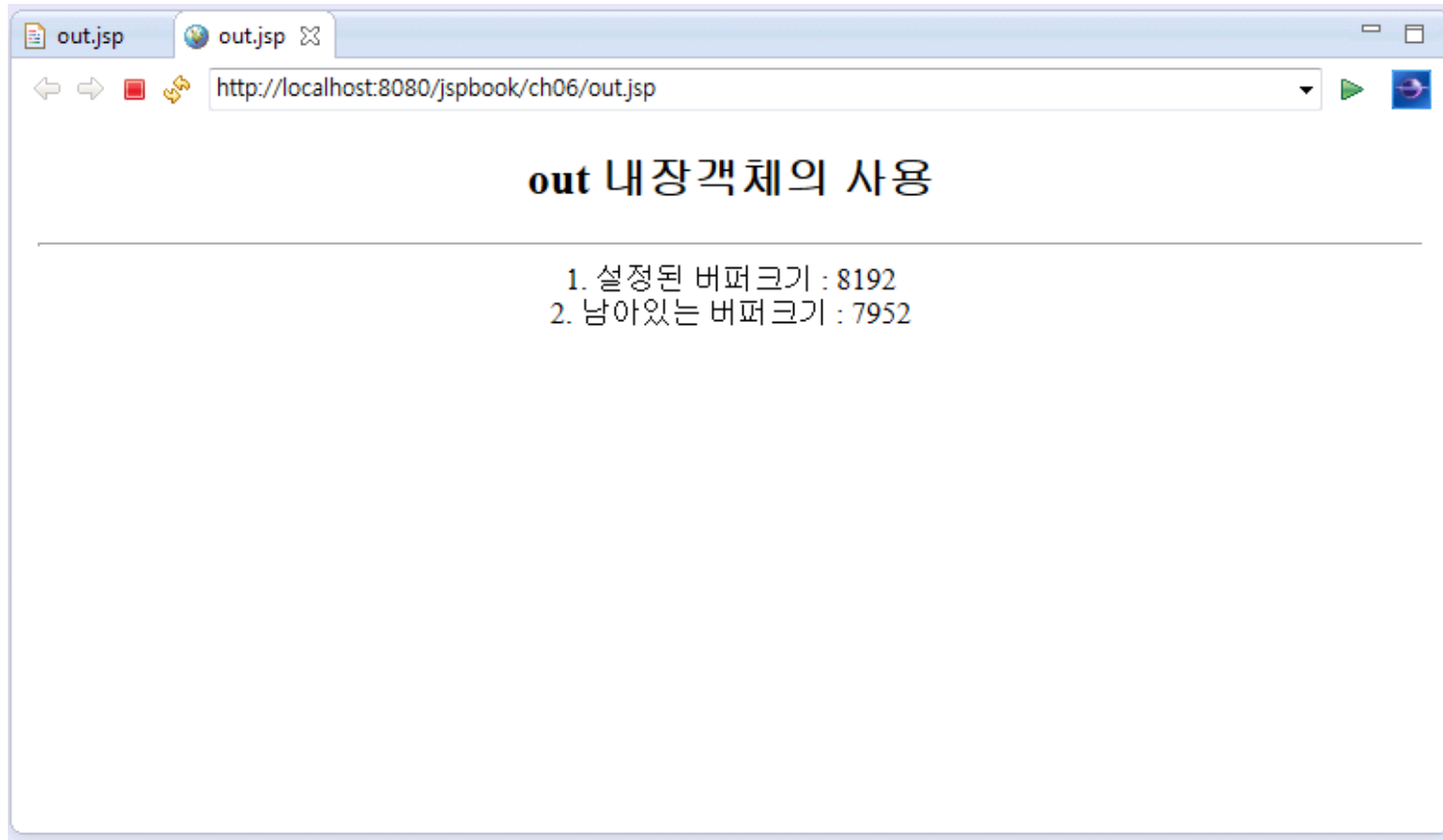
■ out 내장객체

- out은 출력 스트림으로써, 사용자 웹 브라우저로 출력하기 위한 내장 객체임.
- 여러 예제에서 살펴본 것처럼 스크립트릿에서 브라우저로 텍스트를 출력하는 데 사용.
- out은 javax.servlet.jsp.JspWriter 객체의 참조 변수로, 버퍼 관련 메서드와 출력 관련 메서드로 구성되며 out을 이용해서 출력한 내용은 서버의 콘솔이 아닌 사용자에게 전달된다.

메서드	설명
getBufferSize()	output buffer의 크기를 바이트로 알려준다.
getRemaining()	남아 있는 버퍼의 크기 중 사용 가능한 비율을 알려준다.
clearBuffer()	버퍼에 있는 콘텐츠를 모두 지운다.
flush()	버퍼를 비우고 output stream도 비운다.
close()	output stream을 닫고 버퍼를 비운다.
<u>println(content)</u>	content의 내용을 newline과 함께 출력한다.
<u>print(content)</u>	content의 내용을 출력한다.

04. out

■ [실습] out 참고 변수 메서드(out.jsp)



05. session ✱

■ session 내장객체

가

-

.

- HTTP 프로토콜이 비연결형 프로토콜이기 때문에 한 페이지가 출력된 다음에는 클라이언트와 서버의 연결이 끊어진다. 따라서 한번 로그인한 사용자가 로그아웃할 때까지 페이지를 이동해도 보관해야 할 정보가 있다면 이에 대한 처리가 매우 곤란해진다.

- 이러한 HTTP 프로토콜 문제점을 해결하려고 나온 것이 쿠키와 세션이다.

- session 은 javax.servlet.http.HttpSession 인터페이스의 참조 변수 이다.

✱ session 은 접속하는 사용자 별로 따로 생성되며 일정시간 유지되고 소멸된다.

✱ 이러한 세션의 특징을 이용해 setAttribute() 메서드를 이용해 임의의 값을 저장해 놓고 활용할 수 있음.

- 세션이 주로 사용되는 경우는 다음과 같다.

① 사용자 로그인 후 세션을 설정하고 일정 시간이 지난 경우 다시 사용자 인증을 요구 할 때.

② 쇼핑몰에서 장바구니 기능을 구현할 때.

③ 사용자의 페이지 이동 동선 등 웹 페이지 트래킹 분석 기능 등을 구현할 때.

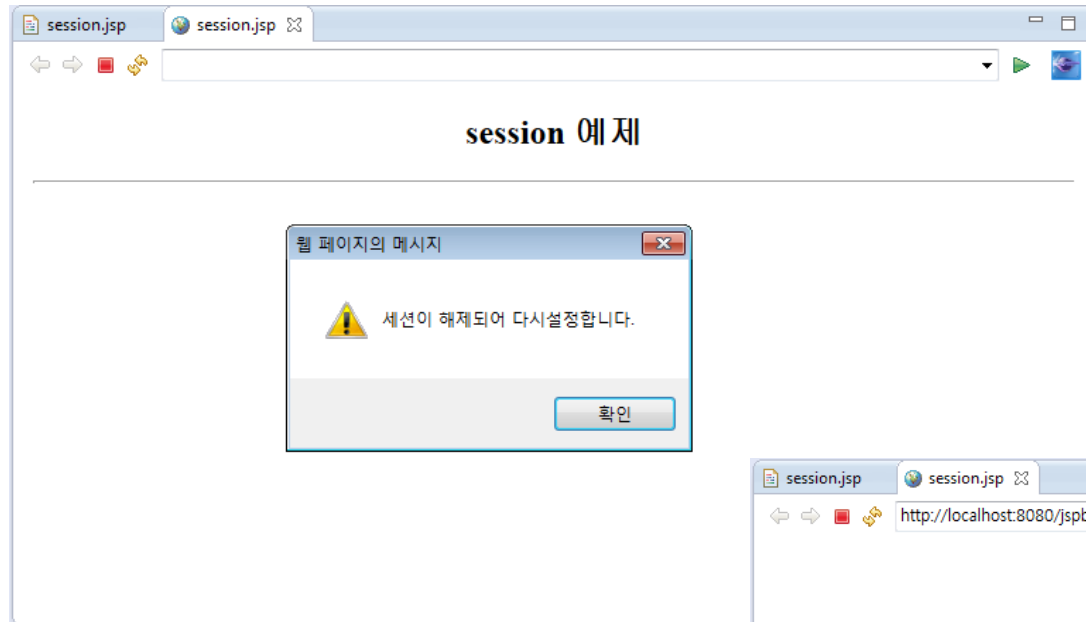
05. session

- session 내장객체의 주요 메서드는 다음과 같다.

메서드	설명
<code>getId()</code>	각 접속에 대한 세션 고유의 ID를 문자열 형태로 반환한다.
<code>getCreatingTime()</code>	세션 생성 시간을 January 1, 1970 GMT.부터 long형 밀리세컨드 값으로 반환한다.
<code>getLastAccessedTime() + 30</code>	현재 세션으로 마지막 작업한 시간을 long형 밀리세컨드 값으로 반환한다.
<code>setMaxInactiveInterval()</code>	세션의 유지 시간을 초로 반환한다. 이를 통해 세션의 유효 시간을 알 수 있다.
<code>setMaxInactiveInterval(t)</code>	세션의 유효 시간을 t에 설정된 초 값으로 설정한다.
 <code>invalidate()</code>	현재 세션을 종료한다. 세션과 관련된 값들은 모두 지워진다.
 <code>getAttribute(attr)</code>	문자열 attr로 설정된 세션 값을 java.lang.Object 형태로 반환한다.
<code>setAttribute(name,attr)</code> 	문자열 name으로 java.lang.Object attr을 설정한다.

05. session

■ [실습] session 내장객체 활용(session.jsp)



06. 그 밖의 내장객체

1. config

- 서블릿이 최초로 메모리에 적재될 때 컨테이너는 서블릿 초기화와 관련된 정보를 읽고 `javax.servlet.ServletConfig` 객체에 저장한다.
- `config`는 바로 `ServletConfig` 클래스에 대한 참조 변수로 `web.xml`에 설정된 초기화 파라미터를 참조하기 위한 용도로 사용할 수 있다.

메서드	설명
<code>getInitParameterNames()</code>	초기 매개변수 값들의 설정 이름을 열거 객체로 반환한다.
<code>getInitParameter(name)</code>	문자열 <code>name</code> 에 해당하는 초기화 매개변수 값을 반환한다.

06. 그 밖의 내장객체

2. application

tomcat

/ tomcat

- application은 웹 애플리케이션(컨텍스트) 전체를 관리하는 객체로 application 객체를 통해 각 서블릿이나 JSP에서 공유하려고 하는 각종 정보를 설정하고 참조할 수 있다.
- application은 ^{data} javax.servlet.ServletContext 객체에 대한 참조 변수로써, config 객체를 통해 생성한다. ServletContext 객체는 컨테이너와 관련된 여러 정보를 제공하며, application 참조 변수를 통해서 서블릿이 실행되는 환경이나 서버 자원과 관련한 정보를 얻거나 로그 파일을 기록하는 작업 등을 수행한다.
- ✱ application 내장객체는 일반적으로 톰캣의 시작과 종료 라이프사이클을 가진다.
- 유형별로 많은 메서드를 제공하므로 주로 관리 기능의 웹 애플리케이션 개발에 유용하다.

06. 그 밖의 내장객체

[표 6-7] 개발자를 위한 서버 정보

메서드	설명
<code>getServerInfo()</code>	JSP/서블릿 컨테이너의 이름과 버전을 반환한다.
<code>getMajorVersion()</code>	컨테이너가 지원하는 서블릿 API의 주 버전 정보를 반환한다.
<code>getMinorVersion()</code>	컨테이너가 지원하는 서블릿 API의 하위 버전 정보를 반환한다.

[표 6-8] 서버 자원 정보

메서드	설명
<code>getMimeType(filename)</code>	문자열 filename에 지정된 파일에 대한 MIME Type을 반환한다.
<code>getResource(path)</code>	문자열 path에 지정된 자원을 URL 객체로 반환한다.

메서드	설명
<code>getResourceAsStream(path)</code>	문자열 path에 지정된 자원을 <code>InputStream</code> 객체로 반환한다.
<code>getRealPath(path)</code>	문자열 path에 지정된 자원을 파일 시스템의 실제 경로로 반환한다.
<code>getContext(path)</code>	문자열 path에 지정된 자원의 컨텍스트 정보를 반환한다.
<code>getRequestDispatcher(path)</code>	문자열 path에 지정된 자원을 위한 request dispatcher를 생성한다.





06. 그 밖의 내장객체

[표 6-9] 로그 관련 정보

메서드	설명
<code>log(message)</code>	문자열 <code>message</code> 의 내용을 로그 파일에 기록한다. 로그 파일의 위치는 컨테이너에 따라 다르다.
<code>log(message,exception)</code>	예외 상황에 대한 정보를 포함하여 로그 파일에 기록한다.

[표 6-10] 속성 관련 정보

메서드	설명
 <code>getAttribute(String name)</code>	문자열 <code>name</code> 에 해당하는 속성 값이 있다면 <code>Object</code> 형태로 가져온다. 따라서 반환 값에 대한 적절한 형변환이 필요하다.
<code>getAttributeNames()</code>	현재 application 객체에 저장된 속성들의 이름을 열거 형태로 가져온다.
 <code>setAttribute(String name, Object value)</code>	문자열 <code>name</code> 이름으로 <code>Object</code> 형 데이터를 저장한다. <code>Object</code> 형이므로 자바 클래스 형태로도 저장할 수 있다.
<code>removeAttribute(String name)</code>	문자열 <code>name</code> 에 해당하는 속성을 삭제한다.

06. 그 밖의 내장객체

3. page

- page는 JSP 컨테이너에서 생성된 서블릿 인스턴스 객체를 참조하는 참조 변수며, JSP에서 자기 자신을 참조할 때 사용된다.
- JSP 스크립트 언어가 자바가 아니라면 유용하게 사용할 수 있지만, 자바인 경우 page 참조 변수를 통하지 않고도 생성된 서블릿 클래스의 멤버변수나 메서드에 직접 접근할 수 있다.
- 따라서 page 참조 변수는 거의 사용하지 않는다.

06. 그 밖의 내장객체

4. pageContext

- pageContext는 javax.servlet.jsp.PageContext 인스턴스에 대한 참조 변수로, 다른 모든 내장객체에 대한 프로그램적인 접근 방법을 제공한다.
- 많이 사용하는 형태는 HTTP 요청을 처리하는 제어권을 다른 페이지로 넘길 때 사용하는 것으로 forward 액션과 동일한 기능을 제공한다.(forward 액션의 내부 구현 코드로 이해할 수 있다.)

06. 그 밖의 내장객체

메서드	설명
<code>getPage()</code>	현재 페이지에서 생성된 서블릿 인스턴스인 <code>page</code> 내장객체를 반환한다.
<code>getRequest()</code>	현재 페이지의 클라이언트 요청 정보가 있는 <code>request</code> 내장객체를 반환한다.
<code>getResponse()</code>	현재 페이지의 클라이언트 응답 정보가 있는 <code>response</code> 내장객체를 반환한다.
<code>getOut()</code>	현재 페이지의 output stream인 <code>out</code> 내장객체를 반환한다.
<code>getSession()</code>	현재 페이지의 <code>session</code> 내장객체를 반환한다.
<code>getServletConfig()</code>	현재 페이지의 <code>config</code> 내장객체를 반환한다.
<code>getServletContext()</code>	현재 페이지의 서블릿 컨텍스트(application 내장객체)를 반환한다.
<code>getException()</code>	오류 페이지, 즉 <code>page</code> 지시어에서 <code>errorPage</code> 속성을 지정한 페이지에서 오류가 발생할 때, 발생한 예외 정보가 있는 <code>exception</code> 내장객체를 반환한다.

06. 그 밖의 내장객체

5. exception

- exception은 page 지시어에서 오류 페이지로 지정된 JSP 페이지에서 예외가 발생할 때 전달되는 `java.lang.Throwable`의 인스턴스에 대한 참조 변수다.
- 이를 통해 현재 페이지를 처리하다 발생하는 예외상황에 대한 정보를 가져올 수 있다.
- 일반적으로 오류 페이지를 별도로 구성하거나 문제 발생할 경우, 로깅을 위한 추가적인 정보를 획득하기 위해 사용한다.

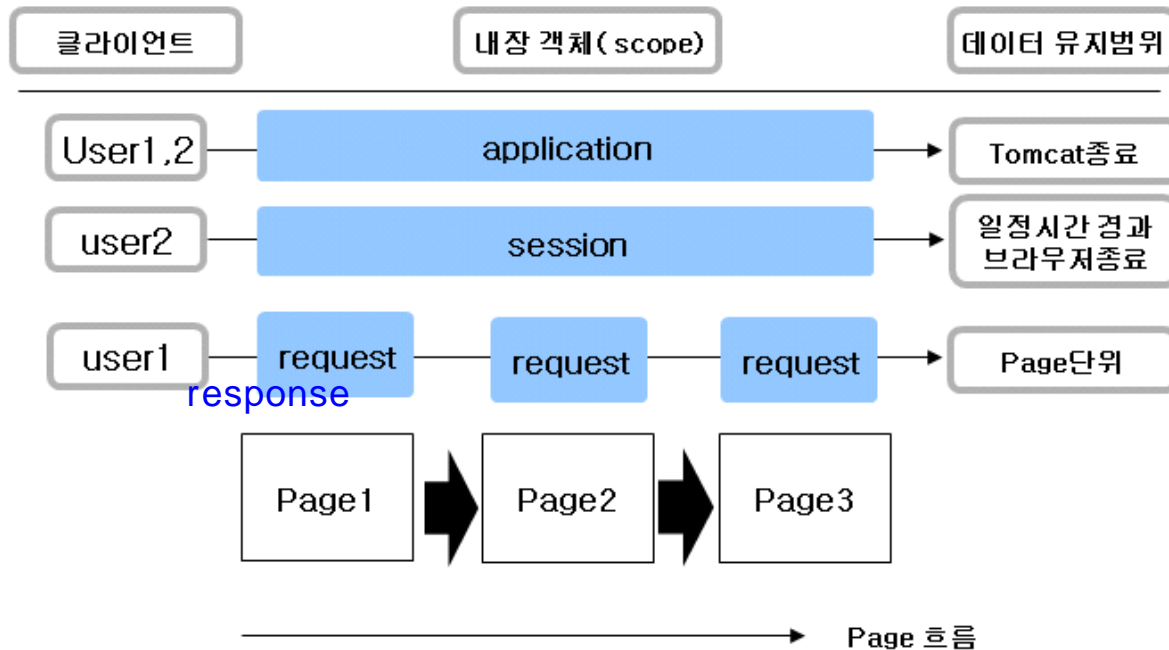
메서드	설명
<code>getMessage()</code>	문자열로 된 오류 메시지를 반환한다.
<code>printStackTrace()</code>	표준 출력 스트림으로써, 스택 추적 정보를 출력한다.
<code>toString()</code>	예외 클래스 이름과 함께 오류 메시지를 반환한다.

07. JSP 내장객체와 속성 관리

1. HTTP 프로토콜 특징과 내장객체 속성 관리

- JSP는 HTTP 프로토콜의 사용하는 웹 환경에서 구동되는 프로그램 이다.
- HTTP는 비연결형으로 사용자가 서버에 특정 페이지를 요청하고 요청결과를 응답받으면 서버와의 연결이 끊기는 형태임.
- 예를 들어 게시판에 글을 작성하는 페이지에서 작성한 내용은 다른 jsp에서 처리해야 하고 서버는 방금 글을 작성한 사람이 누구인지 모를 수 있음.
- 또 다른 예로 쇼핑몰에서 여러 상품 페이지를 이동하면서 장바구니에 물건을 담아 두고 한꺼번에 구매하고자 할 때 접속된 사용자별로 선택된 상품을 처리하는 경우 지금까지 배운 JSP 문법만 가지고는 이를 처리하기 어려움.
- JSP 에서는 page, request, session, application 내장객체를 통해 서로 다른 페이지에서 처리된 값을 저장하고 공유하기 위한 방법을 제공함.
- 이는 컨테이너 기반 프로그램의 특징 중 하나로 실제 프로그램 구현 시 매우 중요한 기법임.

07. JSP 내장객체와 속성 관리



- ❶ application은 모든 사용자가 공유하는 데이터를 저장할 수 있으며 톰캣이 종료될 때 까지 데이터를 유지할 수 있다(맨 위의 user1, user2 해당).
- ❷ session의 경우 사용자마다 분리된 저장 영역이 있으며 Page1, Page2, Page3 모두에서 공유되는 정보를 관리할 수 있다. 물론 이 데이터는 각자 공유 영역에서 관리되며 사용자 간에는 공유되지 않는다.
- ❸ 페이지 흐름이 Page1, Page2, Page3순으로 진행된다고 할 때, 한 페이지에서 다른 페이지로 데이터를 전달하려면 request 내장객체를 이용해야 한다(맨 아래의 user1에 해당한다). page 마다 생성됨.

07. JSP 내장객체와 속성 관리

- request, session, application 은 각각 생성 시점과 소멸시점이 다르며 이를 잘 이해하고 적절한 내장객체를 이용해야 한다.
- 각각의 내장객체는 모두 `getAttribute()`, `setAttribute()` 메서드를 통해 속성을 저장하거나 가져올 수 있다.

[표 6-14] 주요 내장객체의 생성 시점과 소멸 시점

내장객체	생성 시점	소멸 시점
request	해당 페이지 요청 시점	해당 페이지 로딩 완료 시점
session	해당 컨텍스트 내 특정 파일 요청 시점 (사용자 최초 접속 시점)	<ul style="list-style-type: none">• 웹 브라우저 종료 시점• 일정 시간 경과 시점
application	웹 애플리케이션 시작 시점	웹 애플리케이션 종료 시점

07. JSP 내장객체와 속성 관리

2. request, session, application을 이용한 속성 관리

- request, session, application은 **맵 형태의 속성 관리** 기능을 제공 한다.
- 속성을 저장하기 위해서는 `setAttribute(String name, Object value)` 형태를 취한다.
- 반대로 속성에 저장된 값을 가져오는 `getAttribute(String name)` 메서드는 name에 해당하는 Object 를 리턴한다.
- 리턴되는 타입이 Object 이므로 속성을 가지고 올 때에는 적절한 형 변환이 필요하다.
- 예를 들어 page1에서 `session.setAttribute("name","홍길동")`으로 문자열 객체를 저장한다면 page3에서는 `session.getAttribute("name")`으로 저장된 값을 참조할 수 있다.

07. JSP 내장객체와 속성 관리

3. 컨테이너 기반 프로그램의 특징과 JSP 내장객체

- 최신의 프로그램 아키텍처의 특징 중 하나는 컨테이너를 기반으로 한 구조임.
- 프로그램 관점에서 컨테이너란 프로그램 실행에 관여하면서 모듈화된 프로그램을 실행할 수 있게 하고 프로그램 간의 원활한 데이터(객체) 교환을 지원하는 소프트웨어를 말한다.
- JSP에서 내장객체를 이용한 속성 관리가 가능한 것은 JSP와 빈즈 객체들이 톰캣이라고 하는 컨테이너에 의해 관리되고 실행되기 때문이다.
- 웹 프로그램도 컨테이너 기반이며 대표적인 프레임워크인 스프링 역시 컨테이너 기반의 프로그램 모델이 된다. 이러한 컨테이너 기반 프로그램의 장점은 다음과 같다.

- ❶ 프로그램의 모듈화가 용이하다.
- ❷ 독립적으로 실행되는 모듈 간의 데이터 교환이 용이하다.
- ❸ 개별 프로그램에서 화면/상태 전환 시 데이터를 유지/관리하기 용이하다.
- ❹ JSP에서는 컨테이너에 의해 관리되는 내장객체를 통해 임의의 객체를 각각의 생명주기 시점에 따라 공유할 수 있다.

07. JSP 내장객체와 속성 관리


4. MVC 패턴과 JSP 내장객체

- MVC 패턴은 프로그램을 Model, View, Controller 세가지 역할로 구분해 구현하는 소프트웨어 디자인 패턴을 말함. , ,view
- MVC 패턴에 따르면 JSP는 뷰의 역할만 수행해야 함. 즉 화면에 데이터를 출력하는 기능만 제공해야 한다는 것인데, 문제는 컨트롤러에서 처리한 데이터(예를 들면 데이터베이스로부터 가져온)를 어떻게 JSP로 전달해야 하는지에 대한 것이다.
- 이처럼 MVC 패턴을 사용하게 되면 JSP는 별도로 데이터를 가지고 오는 로직 없이 데이터를 출력할 수 있어야 하는데 이때 JSP 내장객체를 이용한 속성 관리가 사용된다.
- 예를 들어, 컨트롤러에서 처리한 데이터는 `request.setAttribute()`를 이용해서 저장하고 화면에 보여질 JSP로 포워딩 하면 해당 JSP에서는 request 내장객체를 통해 데이터를 참조할 수 있으므로 완전한 MVC 패턴의 구현을 구현할 수 있다.
- 뷰를 효과적으로 구성하는 방법은 `<jsp:useBean>`, `<jsp:getProperty>`, 표현식을 사용하는 것 이나 표현 언어(Expression Language)와 JSTL을 이용할 경우 더욱 편리하게 뷰를 구현할 수 있다.

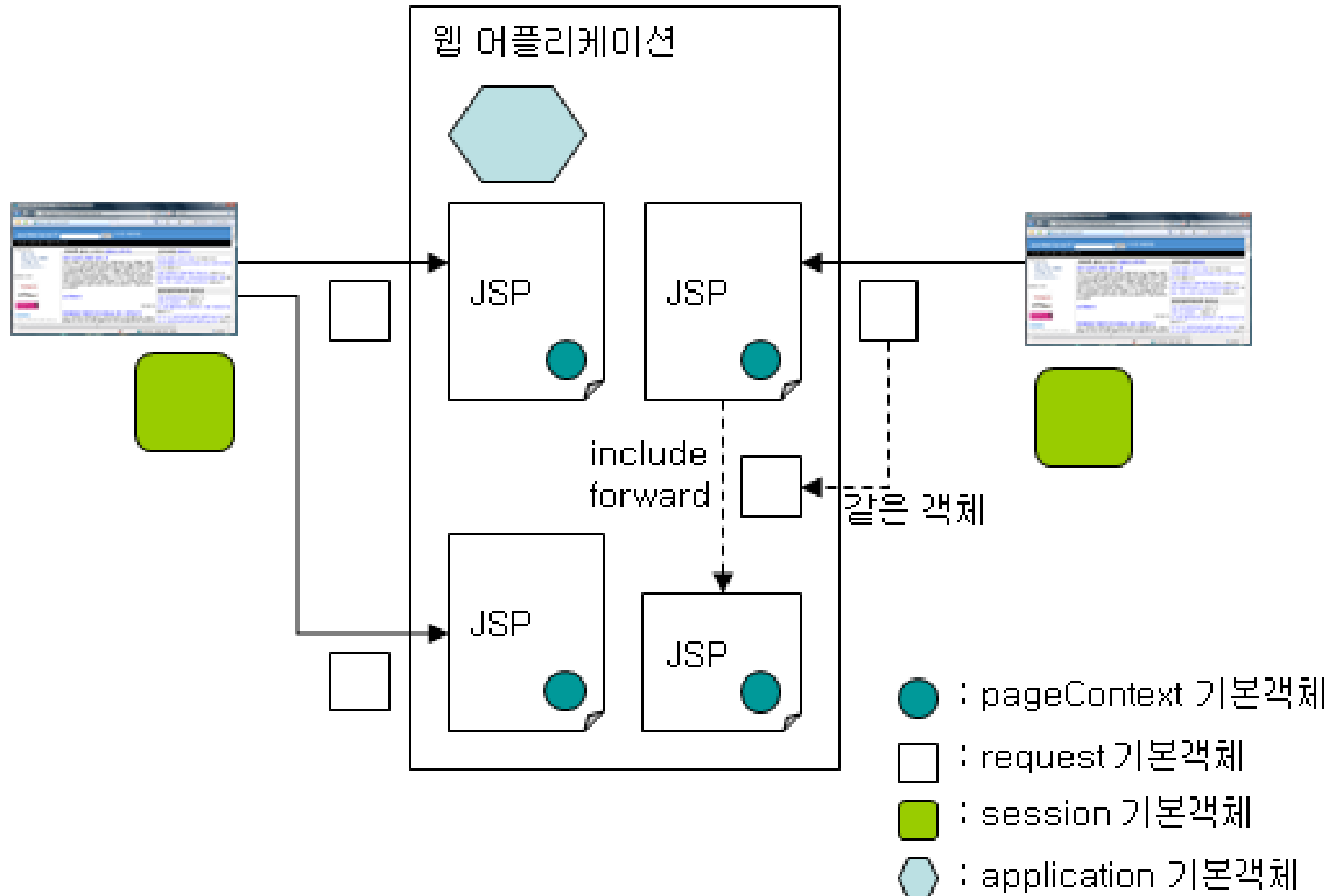
■ 기본 객체와 영역

- 네 영역

spring

- 
- PAGE 영역 - 하나의 JSP 페이지를 처리할 때 사용되는 영역
 - REQUEST 영역 - 하나의 HTTP 요청을 처리할 때 사용되는 영역
 - SESSION 영역 - 하나의 웹 브라우저와 관련된 영역
 - APPLICATION 영역 - 하나의 웹 어플리케이션과 관련된 영역

■ 기본 객체와 영역



■ 속성(Attribute)

- 속성 기능 제공 기본 객체
 - `pageContext`, `request`, `session`, `application`
- 속성 관련 메서드

메서드	리턴타입	설명
<code>setAttribute(String name, Object value)</code>	<code>void</code>	이름이 <code>name</code> 인 속성의 값을 <code>value</code> 로 지정한다.
<code>getAttribute(String name)</code>	<code>Object</code>	이름이 <code>name</code> 인 속성의 값을 구한다. . 지정한 이름의 속성이 존재하지 않을 경우 <code>null</code> 을 리턴한다.
<code>removeAttribute(String name)</code>	<code>void</code>	이름이 <code>name</code> 인 속성을 삭제한다.
<code>getAttributeNames()</code>	<code>Enumeration</code>	속성의 이름 목록을 구한다. (<code>pageContext</code> 기본 객체는 이 메서드를 제공하지 않는다.)

■ 속성(Attribute)

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%      String name = request.getParameter("name");
        String value = request.getParameter("value");

        if (name != null && value != null) {
            application.setAttribute(name, value);
        }
%>
<html>
<head><title>application 속성 지정</title></head>
<body>
<%      if (name != null && value != null) {    %>
application 기본 객체의 속성 설정:
    <%= name %> = <%= value %>
<%      } else {    %>
application 기본 객체의 속성 설정 안 함
<%      }          %>
</body>
</html>
```

■ 속성(Attribute)

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ page import = "java.util.Enumeration" %>
<html>
<head> <title>application 기본 객체 속성 보기</title> </head>
<body>
<%
    Enumeration attrEnum = application.getAttributeNames();
    while(attrEnum.hasMoreElements() ) {
        String name = (String)attrEnum.nextElement();
        Object value = application.getAttribute(name);
%>
application 속성 : <b><%= name %></b> = <%= value %> <br>
<%
    }
%>
</body>
</html>
```

■ 속성의 활용

기본 객체	영역	쓰임새
pageContext	PAGE	(한번의 요청을 처리하는) 하나의 JSP 페이지 내에서 공유될 값을 저장한다.
request	REQUEST	한번의 요청을 처리하는 데 사용되는 모든 JSP 페이지에서 공유될 값을 저장한다.
session	SESSION	한 사용자와 관련된 정보를 JSP 들이 공유하기 위해서 사용된다.
application	APPLICATION	모든 사용자와 관련해서 공유할 정보를 저장한다.

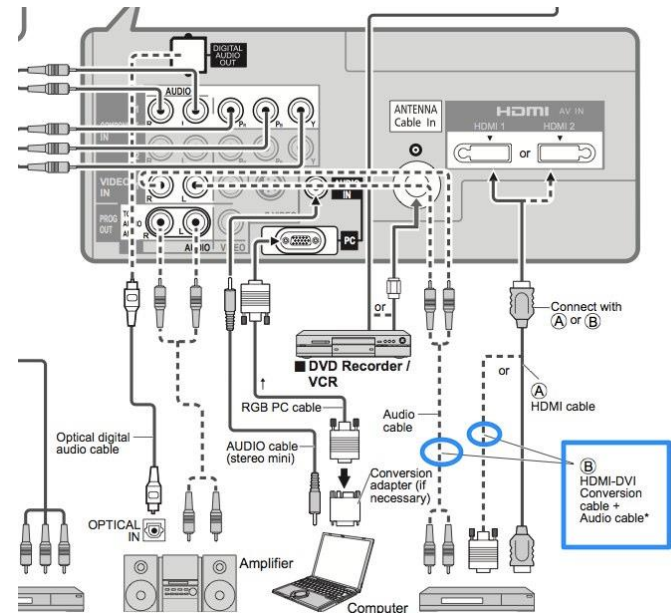
■ 빈즈 개요

- 빈즈란 ?

- ^{- >} 빈즈(Beans)는 특정한 일을 독립적으로 수행하는 컴포넌트를 의미한다.
- 원래 자바에서는 GUI(Graphic User Interface), 즉 창이나 버튼, 스크롤바 등 화면을 구성하는 다양한 위젯을 제작하려고 빈즈를 만들었다.
- J2EE가 발표되면서 각각 엔터프라이즈 자바 빈즈(EJB : Enterprise Java Beans)와 JSP에서 사용하는 JSP 빈즈로 나뉘어 개념이 확장되었다. 이들의 용도는 다르지만 프로그램 모듈화를 위한 컴포넌트라는 기본 전제는 같다.

1. 자바 빈즈

- 일반적으로 컴포넌트라고 하면 다른 무언가를 만들기 위한 부품을 말한다. 컴포넌트는 각각 독립적인 기능이 있으며, 컴포넌트 조합을 통해 다양한 형태의 결과물을 만들 수 있다. 예를 들어, 레고 블록이나 예전의 컴포넌트 오디오 등을 생각해볼 수 있다.
- 이때 각각의 모듈을 서로 조합하려면 규격화된 인터페이스가 있어야 한다. 레고의 경우에는 튀어나온 부분들이 다른 블록의 아래쪽에 결합이 되는 구조고, 컴포넌트 오디오는 일반 스테레오 케이블이나 HDMI, 광케이블 등으로 서로 연결된다.



2. JSP 빈즈

- JSP 빈즈는 JSP와 연동하기 위해 만들어진 컴포넌트 클래스이다.
- 컨테이너에 위치하며, JSP에 데이터베이스 연동 등 프로그램적 요소를 모듈화할 수 있도록 도와준다.
- 데이터 처리와 공용화된 기능을 제공(VO, DTO)하기 때문에 빈즈를 잘 활용하면 프로그램의 중복을 줄이고 더욱 원활하게 유지·보수할 수 있다.
- 따라서 가능하면 JSP 코드 내에 스크립트릿을 사용하는 것보다는 빈즈를 만들어 사용하는 것이 좋다.

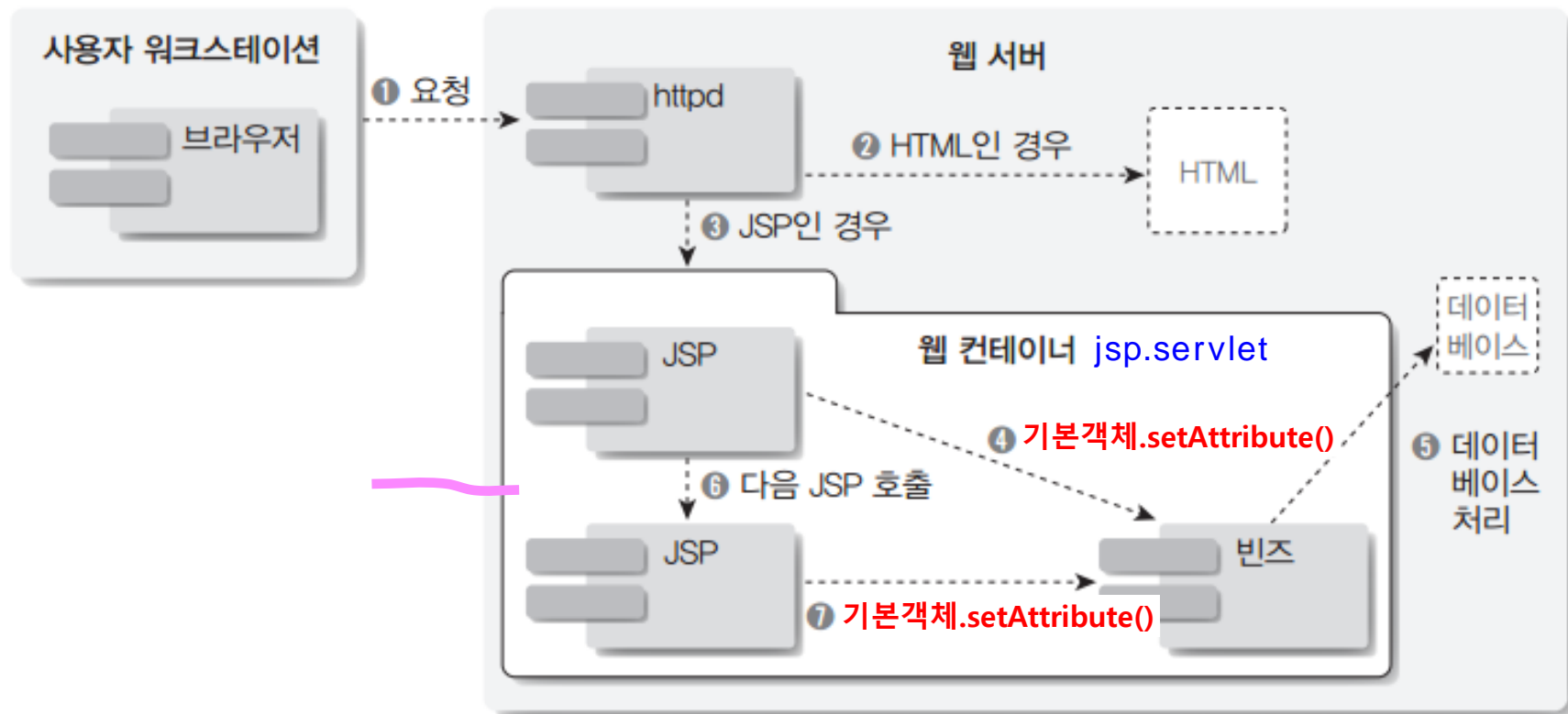
VO : Value Object

DTO : Data transfer objec

```
<jsp : usebean
```

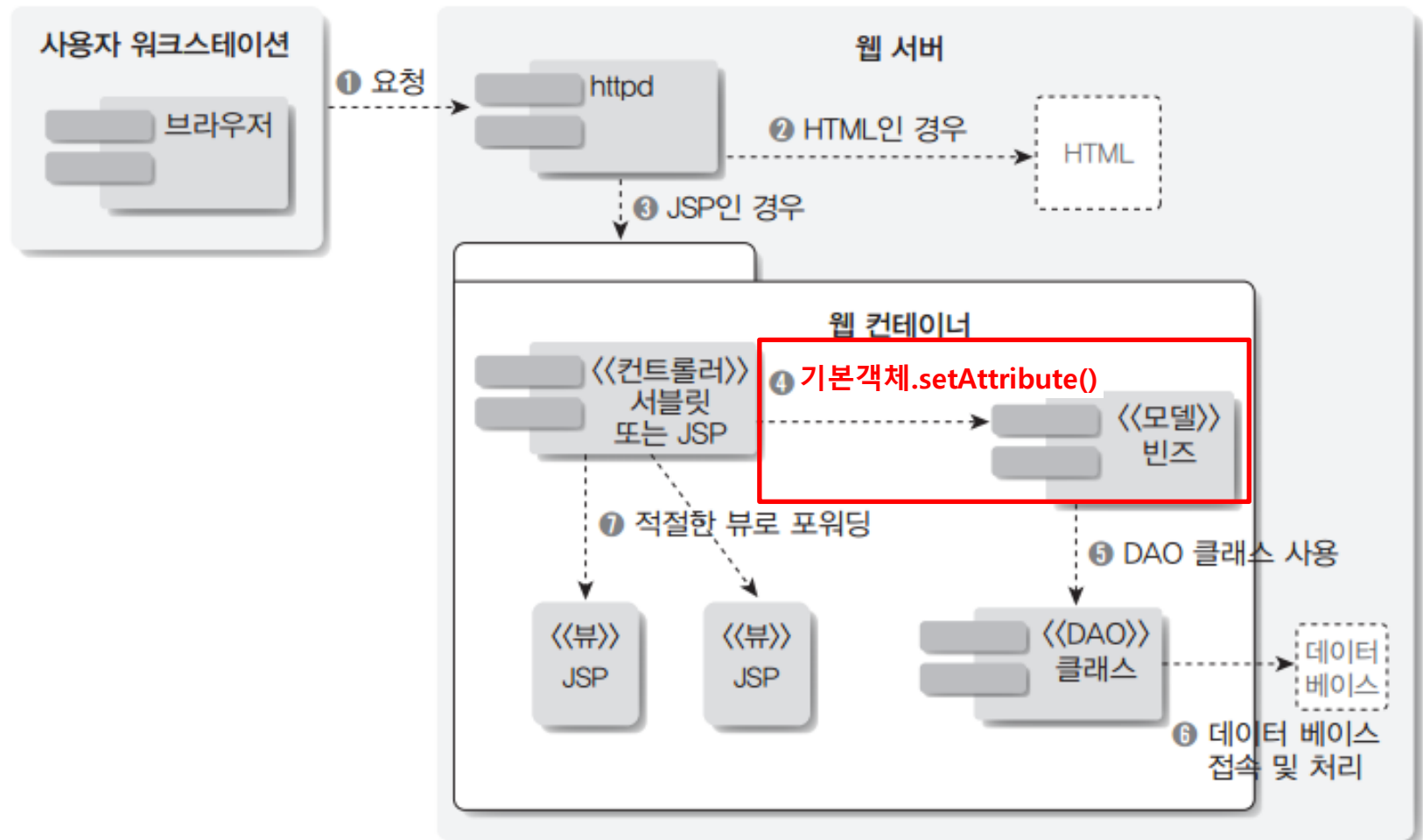
```
(tomcat . )
```

2. JSP 빈즈



■ 빈즈 개요

- MVC 패턴에 기반해 프로그램 개발시에는 개별 JSP에서 빈즈에 접근하는 것 보다는 컨트롤러에서 빈즈와 연동하고 request, session, application 등 내장객체의 속성 관리 기능을 이용해 해당 뷰(JSP)에 빈즈 객체를 전달하는 방법이 권장된다.



■ JSP와 빈즈 연동

빈즈 연동

spring+mvc

- JSP빈즈는 JSP에서 사용할 수 있는 자바 컴포넌트로 빈즈 액션과 결합해 웹 프로그램을 더욱 간편하고 단순한 구조로 개발 할 수 있게 해준다.
- 빈즈도 자바 클래스이기 때문에 JSP 파일의 스크립트릿 부분에서 인스턴스를 만들어 사용할 수 있다. 그러나 일반 클래스와 다를 바 없다면 굳이 빈즈라는 이름을 붙일 필요가 없을 것이다.
- 빈즈는 일반 클래스처럼 사용할 수도 있지만, JSP 빈즈만의 특징적인 요소를 잘 활용하면 더욱 편하게 프로그램을 작성할 수 있다.
- 일반적으로 빈즈를 사용할 경우, 사용자 입력 처리에 필요한 HTML이나 JSP와 폼의 액션에 연결된 JSP 파일이 있어야 한다. 또한 폼을 처리하는 JSP 파일에서 참조할 빈즈 클래스도 필요하다.
- 물론 경우에 따라 결과를 보여주는 HTML이나 JSP 파일이 필요할 수도 있다.

~~new member();~~

■ 자바빈(JavaBeans)

- 자바빈 - 웹 프로그래밍에서 데이터의 표현을 목적으로 사용
- 회원정보, 게시판 등의 정보 출력
- 일반적인 구성
 - 값을 저장하기 위한 필드
 - 값을 수정하기 위한 setter
 - 값을 읽기 위한 getter

```
public class BeanClassName {  
    /* 값을 저장하는 필드 */  
    private String value;  
  
    /* BeanClassName의 기본 생성자 */  
    public BeanClassName() {  
    }  
  
    /* 필드의 값을 읽어오는 값 */  
    public String getValue() {  
        return value;  
    }  
  
    /* 필드의 값을 변경하는 값 */  
    public void setValue(String value) {  
        this.value = value;  
    }  
}
```

■ 자바빈(JavaBeans)

빈즈 클래스 구조

- 기본적으로 빈즈 클래스는 자바 클래스이므로, 일반적인 자바 클래스 구성을 따른다.
- JSP 액션과 연동하기 위해 필요한 몇 가지 필수 구성의 차이가 있지만 이는 문법적인 제약이 아니기 때문에 규칙을 따르지 않더라도 컴파일 오류가 발생하지는 않는다.
- 빈즈는 매개변수가 없는 기본 생성자를 요구하므로 만일 매개변수가 있는 생성자를 구현하였다면 기본 생성자를 명시적으로 선언해 주어야 문제가 발생하지 않는다.
- 빈즈 클래스의 일반적인 구조는 **멤버변수, getter/setter 메서드** 이다.

```
class xxxBean {  
    // 멤버변수 : 데이터베이스 테이블의 칼럼 이름과 매칭된다.  
    private String xxx;  
  
    // get, set 메서드 : 멤버변수와 매칭된다.  
    public String getXxx() {  
        return xxx;  
    }  
    public setXxx(String xxx) {  
        this.xxx = xxx;  
    }  
}
```

■ 자바빈(JavaBeans)

```
class xxxBean {  
    // 멤버변수 : 데이터베이스 테이블의 칼럼 이름과 매칭된다.  
    private String xxx;  
  
    // get, set 메서드 : 멤버변수와 매칭된다.  
    public String getXxx() {  
        return xxx;  
    }  
    public setXxx(String xxx) {  
        this.xxx = xxx;  
    }  
}
```


■ 자바빈 프로퍼티

- 프로퍼티는 자바빈에 저장되어 있는 값을 표현
- 메서드 이름을 사용해서 프로퍼티의 이름을 결정
- 규칙 : 프로퍼티 이름이 propertyName일 경우
 - setter: `public void setPropertyName(Type value)`
 - getter: `public Type getPropertyName()`
 - `boolean` 타입일 경우 getter에 `get`대신 `is` 사용 가능
 - 배열 지정 가능: 예) `public void setNames(String[])`
- 읽기/쓰기
 - 읽기 전용 : `get` 또는 `is` 메서드만 존재하는 프로퍼티 VO
 - 읽기/쓰기 : `get/set` 또는 `is/set` 메서드가 존재하는 프로퍼티

src/member/MemberInfo.java

```
package member;
```

```
import java.util.Date;
```

```
public class MemberInfo {
```

```
    private String id;
```

```
    private String password;
```

```
    private String name;
```

```
    private String address;
```

```
    private Date registerDate;
```

```
    private String email;
```

```
    public String getId() {
```

```
        return id;
```

```
    }
```

beans

DB

HTML FORM

src/member/MemberInfo.java

```
public void setId(String val) {  
    this.id = val;  
}  
  
public String getPassword() {  
    return password;  
}  
  
public void setPassword(String val) {  
    this.password = val;  
}  
  
public String getName() {  
    return name;  
}  
  
public void setName(String val) {  
    this.name = val;  
}
```

src/member/MemberInfo.java

```
public String getAddress() {  
    return address;  
}  
  
public void setAddress(String val) {  
    this.address = val;  
}  
  
public Date getRegisterDate() {  
    return registerDate;  
}  
  
public void setRegisterDate(Date val) {  
    this.registerDate = val;  
}  
  
public String getEmail() {  
    return email;  
}  
  
public void setEmail(String val) {  
    this.email = val;}}
```